**Aim:** Implement a program on method and constructor overloading.

**Objective:** To use concept of method overloading in a java program to create a class with same function name with different number of parameters.

## Theory:

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

Example: This example to show how method overloading is done by having different number of parameters for the same method name.

import java.io.\*;

class Student {

public void StudentId(String name, int roll\_no){

System.out.println("Name :" + name + " "

+ "Roll-No :" + roll\_no);

}

public void StudentId(int roll\_no, String name)

{

// Again printing name and id of person

System.out.println("Roll-No :" + roll\_no + " "

+ "Name :" + name);

}

}

class Stdetails {

public static void main(String[] args)

{

Student obj = new Student();

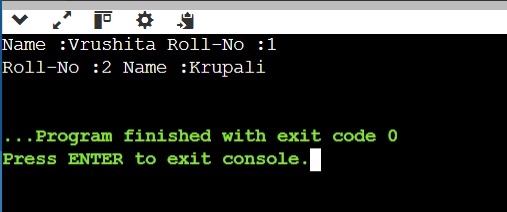
obj.StudentId("Vrushita", 1);

obj.StudentId(2, "Krupali");

}

}

**Output:**



**Constructor Overloading:**

**Code:**class Rectangle

{

int length, width;

void getData(int x, int y)

{

length=x;

width=y;

}

int rectArea()

{

int area=length\*width;

return area;

} }

class demo1

{

public static void main(String args[])

{

int area1,area2;

Rectangle rect1=new Rectangle();

Rectangle rect2=new Rectangle();  
 rect1.length=20;

rect1.width=30;

area1=rect1.length\*rect1.width;

rect2.getData(20,10);

area2=rect2.rectArea();

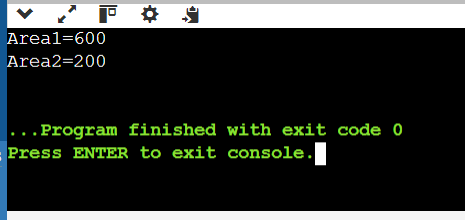
System.out.println("Area1="+area1);

System.out.println("Area2="+area2);

}

}

**Output:**

****

**Conclusion:**Overloading simplifies method naming conventions by using the same name for related methods. This enhances code readability as developers can easily understand the purpose of these methods. method overloading is a valuable programming technique that enhances code readability, maintainability, and flexibility by allowing multiple methods with the same name to exist in a class, differing only in their parameter lists. It simplifies code development, improves code reuse, and contributes to a more expressive and user-friendly programming experience.